IN THE DRAWINGS

The Examiner objected to the drawings under 37 CFR 1.83(a). The objection to the drawings is address in the Remarks section of the present response.

REMARKS

Rejection Under 35 U.S.C. 112(1) And Objection To Drawings Under 37 CFR 1.83(a)

The Examiner rejected claims 1-30 under 35 U.S.C. 112(1). The Examiner argues that no representative instruction and cache implementation detailing how a reset may initiated an instruction sequence which may itself invalidate a sequence of cache lines in lieu of an alternative hardware based mechanism. Along a similar vein, the Examiner objected to the drawings because the Examiner believes that they do not show a software instruction based cache line invalidate implementation versus a more conventional hardware based mechanism. The Applicants respectfully disagree.

As the Examiner recognizes, available mechanisms for bypassing a processor cache are well known. In some examples, multiplexers are used to bypass a processor cache. "Providing a mechanism to bypass the processor cache uses extra hardware resources. In general-purpose processors or custom application-specific integrated circuits (ASICs), the extra hardware resources used can be efficiently implemented, although the techniques of the present invention can still be applied to allow for other benefits. In programmable chips, the extra hardware resources needed to bypass the cache are not trivial. Mechanisms such as multiplexers used for bypass circuitry on a programmable chip are relatively expensive. (page 7, lines 12-18; Figure 4)

Consequently, "the techniques of the present invention allow reset handling without bypass circuitry. According to various embodiments, the techniques of the present invention provide a processor with specialized circuitry or hardware to invalidate the reset address line associated with an instruction cache. The techniques of the present invention recognize that providing specialized circuitry or hardware to invalidate all lines in both an instruction and a data cache would be substantial, particularly since a reset is a relatively uncommon event. Consequently, mechanisms are provided for using hardware to invalidate a reset address line and instruction cache while allowing software routine to invalidate the other lines in both the instruction and data cache." (page 7, line 27 – page 8, line 4)

Available mechanisms for invalidating a cache line are well known. In one example, invalidating a cache line involves writing an invalid bit to a particular cache line during power

up upon reset. Mechanisms for detecting a reset or reboot to trigger writing of an invalid bit may be the same mechanisms used to detect a reset or reboot to trigger bypassing a cache under conventional mechanisms. "In some examples, it may be possible to specially configure circuitry to invalidate all the lines of a cache 531 including all instruction cache lines and all data cache lines. However, providing such specialized circuitry can be resource intensive, particularly for programmable chips. Consequently, the techniques of the present invention envision providing specialized circuitry to invalidate a limited number of lines and instruction cache 531. In one example, a single line including several instructions is invalidated. The several instructions are then used to initiate a software routine to invalidate the remaining lines in the cache 531. In one example, the software routine invalidates only the instruction cache lines. In another example, lines including both instruction cache lines in data cache lines are invalidated. Any logic or mechanism used to invalidate a reset address line is referred to herein as reset address line invalidate circuitry." (page 11, lines 4-15, Figure 5, Figure 6)

According to various embodiments, instead of using conventional mechanisms for bypassing a possibly corrupt cache upon reset, e.g. Figure 4, reset line invalidation circuitry is used instead. Consequently, in some embodiments, the cache is always enabled. According to various embodiments, "at 605, specialized circuitry associated with a processor is used to invalidate the reset address line in the instruction cache. According to various embodiments, a single line in instruction cache is invalidated. However, it should be recognized that multiple lines can also be invalidated using specialized circuitry. At 609, a processor performs a read access for the instruction. At 611, processor recognizes the cache miss based on the invalid state of the reset address line. Because of the invalid state, the processor obtains instructions from memory at 613. According to various embodiments, the instructions are obtained from memory to initialize a software routine that invalidates other instruction cache lines at 617. At 618, the software subroutine invalidates data cache lines. At 621, instructions and data are obtained from memory after cache misses based on the invalid state of the various instruction cache and data cache lines. A reset handler effectively allows a processor to return to a known state. (page 12, lines 1-9)

Consequently, because one of skill in the art would be able to configure a processor and cache to invalidate, e.g. writing an invalid bit to, one or more cache lines upon detecting a reset or reboot and because one of skill in the art would be able to provide a software routing to invalidate other cache lines at the memory line corresponding to the invalidated cache line

upon reviewing the specification and drawings, the 35 U.S.C. 112(1) rejection and the objection to the drawings are believed overcome.

Rejection Under 35 U.S.C. 102(b)

The Examiner rejected independent claims 1, 13, 24, and 28 under 35 U.S.C. 102(b) as being anticipated by Paysan ("A Four Stack Processor" April 2000).

Paysan notes "exceptions and interrupts use the same call mechanism: the first four instructions are fetched out of the exception table and the memory register set switches to the interrupt memory register set. These four instructions will be executed and are guaranteed not to be interrupted. If they cause an exception, the CPU state is lost, so be carefully! If your code isn't exception free, form a long exception by calling the dangerous code part." (page 21, paragraph 4)

However, Paysan does not teach or suggest "wherein a reset address line associated with the instruction cache is invalidated using reset address line invalidate circuitry upon reset." Paysan does not describe invalidating cache lines or handling resets. In fact, Paysan only mentions the term "invalidate" a single time in the entire document during a reference to a multiprocessor protocol. (page 12, paragraph 6) Paysan does not describe invalidating a cache line upon reset and in fact provides no description on how resets are handled. It is respectfully submitted that Paysan likely uses conventional mechanisms for handling resets, i.e. conventional mechanisms that allow a cache to be bypassed entirely. Paysan merely notes that "these four instructions will be executed and are guaranteed not to be interrupted." (page 21, paragraph 4) It should be recognized that these four instructions can be executed after a conventional cache bypass mechanisms is used.

CONCLUSION

In light of the above remarks relating to the independent claims, the remaining dependent claims are believed allowable for at least the reasons noted above. Applicants believe that all pending claims are allowable and respectfully request a Notice of Allowance for this application from the Examiner. Should the Examiner believe that a telephone conference would expedite the prosecution of this application, the undersigned can be reached at the telephone number set out below.

Respectfully submitted,
BEYER WEAVER & THOMAS, LLP

Godfrey K. Kwan
Reg. No. 46,850

P.O. Box 70250
Oakland, CA 94612-0250
(510) 663-1100